ALS timing system upgrade: MRF Embedded EVR

Carlos Serrano, ATG@LBNL

February 7, 2014

Contents

1 MRF Embedded EVR

This package is based on the BNL's MRF embedded receiver design, which is targeted for a Xilinx Virtex-6 LX240T-FF1156 for use in the NSLS-II BPM. We (LBNL) have re-packaged the interaction with the GTX transceiver to include:

- A clean separation between MRF specific and chip/board specific logic,

  This enables easy re-use of the MRF protocol logic using different FPGA/Board targets.

- GTX wrapper and configuration changes.

  Provides a deterministic behavior, where the phase between the RF source at the transmitter end and the recovered clocks at the receiver end are kept constant, even after a loss of lock, firmware re-load or power cycle.

These changes were needed since LBNL intends to use the MRF Embedded EVR design to provide orbit clocks for the BPM ADCs using a divided version the the recovered RF clock. The transmitter end is clocked using the revolution frequency divided by 4 (∼124.913

MHz), which is then multiplied by 20 to serialize the data on the high-speed optical fiber link. At the receiver end, a Clock Data Recovery (CDR) mechanism is used to recover the clock from the data inside the FPGA GTX transceiver. The data is parallelized and the recovered parallel clock (back into the ∼125MHz regime) is provided to the FPGA fabric to clock the incoming parallel data. Since the transmitter end used the RF clock to clock the data, this clock is then RF synchronous and can be used to clock devices such as the BPMs in the storage ring. In particular it is divided by 82 (500MHz/328 or 125MHz/82) to obtain the Storage Ring Orbit Clock (SROC). The changes included here guarantee that the transceiver will provide clocks (either the RF/4 or a derivative) with a fixed phase relationship with the RF even after a loss of lock, firmware re-load or power cycle (both on the transmitter and receiver end).

Since the use of the embedded receiver logic would be desirable in any other FPGA system with a optical fiber link for on-chip integration with the timing system, we have separated the MRF specific logic from the HW specific logic so that the design can be easily re-used using a different FPGA family or vendor.

1.1 Package includes

- MRF Embedded EVR logic (HW independent),
- Deterministic GTX instantiation for a Xilinx Virtex-6 FPGA,
- MRF EVR instantiation example, along with minor logic needed around it.

1.2 Design hierarchy (including changes from LBNL)

- `EventReceiverTop.v`

  This is the top level module for the MRF Embedded EVR. Slight changes were made to this module, but the interface is kept compatible with BNL's BPM project: Added some diagnostic outputs for comma detection and loss of lock, which can be left unconnected if not used.

- `EventReceiverTop_inst_example.v`

  Example instantiation of EventReceiverTop. This module is intended for illustration purposes only, although it could be integrated into a larger design if needed (it has not been tested in HW). It includes some minor external logic needed for the ALS test stand for integration with the BPM project. The interface of `EventReceiverTop_inst_example` corresponds to FPGA pins directly (indicated for illustration purposes only). It does not include buffering of the recovered clock signals to drive differential pairs, which would be desirable (rather than the current single-ended scheme) when integrated with the BPM to clock the ADCs on the AFE.

- `serialReceiver.v`

  Separated the HW specific logic from the MRF protocol logic to enable portability. This module includes some MRF specific logic and instantiates `gtx_wrapper_v6.v`, which contains the HW specific logic.

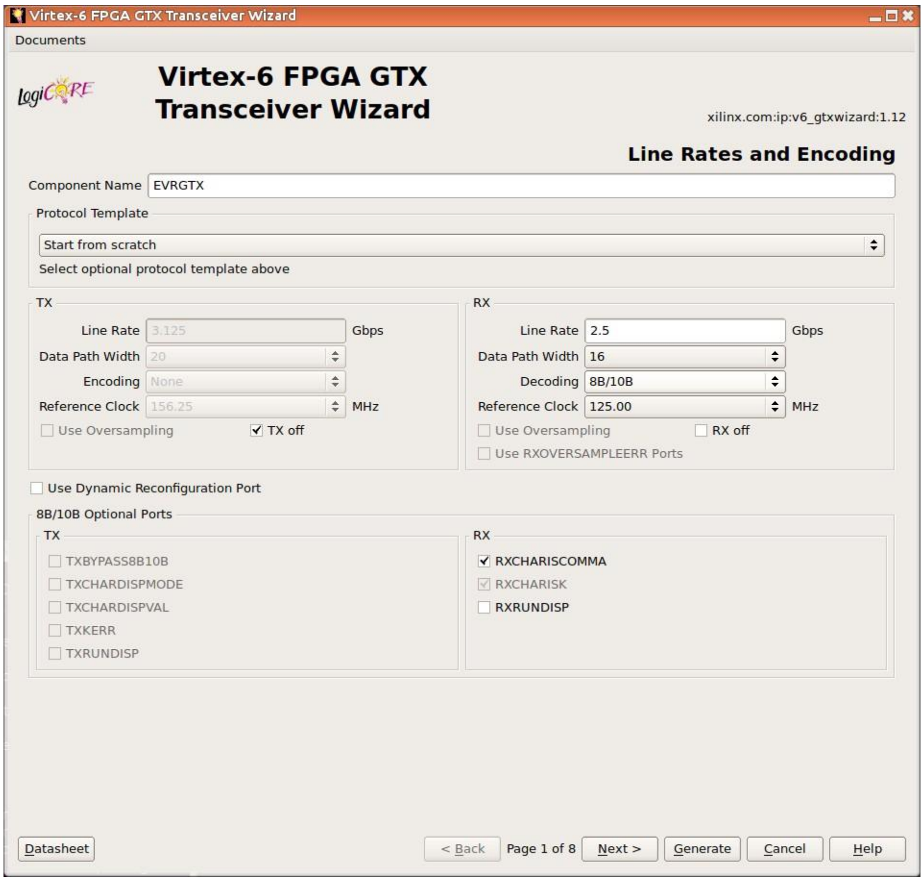
- `gtx_wrapper_v6.v`

  This module is a wrapper on a wrapper on a wrapper (literally) of the GTX transceiver instance in the Virtex-6 FPGA. The main purpose of the module is to provide the translation between the serial received high-speed data (2.5 Gbps) into a 16-bit parallel word clocked at 20 times lower the speed. Only the receiver path is currently supported since the MRF receivers do not use their transmitter path. This module also provides the ~125MHz clock recovered using the CDR in the transceiver and guarantees that its phase is kept constant with that of the RF source used at the transmitter end even after a loss of lock, firmware re-load or power cycle.

- `gtp_bitslide.v, gtx_byteisaligned.v, gtx_rx_phase_force.v`

  These three modules are interconnected in `gtx_wrapper_v6.v` to guarantee deterministic behavior of the recovered parallel clock on the GTX transceiver receiver channel. The Xilinx transceiver Wizard was used to generate a GTX wrapper (`evrgtx.v` and `evrgtx_gtx.v`) which is configured to receive data at 2.5 Gbps and provide a 16-bit data interface along with the recovered parallel clock at ~125MHz. The manual bitslide feature was enabled (see GTX Wizard settings in the next section) since using this mechanism is the only way to detect and reinforce the serial data latency through the receiver logic. `gtp_bitslide` contains a state machine that will slide the serial stream until parallel word alignment is found (indicated by `gtx_byteisaligned.v`). The number of bits that the serial stream needed to be shifted by in order to achieve parallel word alignment indicates the phase relationship between the source RF at the receiver end and the recovered clock. There is no way to reinforce this but by resetting the transceiver and repeating the process until a desired value is found (handled by `gtx_rx_phase_force.v`). The desired value is arbitrary as far as the BPM is concerned since phase alignment of the data can be done using beam-based measurements, but it is important that the phase with the RF is kept constant even after a power cycle, loss of link or re-programming of the FPGA.

- `evr_gtx.v, evr_gtx_gtx.v`



Virtex-6 FPGA GTX Transceiver Wizard

xilinx.com:ip:v6_gtxwizard:1.12

Line Rates and Encoding

Component Name: EVRGTX

Protocol Template: Start from scratch
Select optional protocol template above

TX		RX	
Line Rate	3.125 Gbps	Line Rate	2.5 Gbps
Data Path Width	20	Data Path Width	16
Encoding	None	Decoding	8B/10B
Reference Clock	156.25 MHz	Reference Clock	125.00 MHz
☐ Use Oversampling	✓ TX off	☐ Use Oversampling	☐ RX off
		☐ Use RXOVERSAMPLEERR Ports	

☐ Use Dynamic Reconfiguration Port

8B/10B Optional Ports

TX	RX
☐ TXBYPASS8B10B	✓ RXCHARISCOMMA
☐ TXCHARDISPMODE	✓ RXCHARISK
☐ TXCHARDISPVAL	☐ RXRUNDISP
☐ TXKERR	
☐ TXRUNDISP	

Page 1 of 8

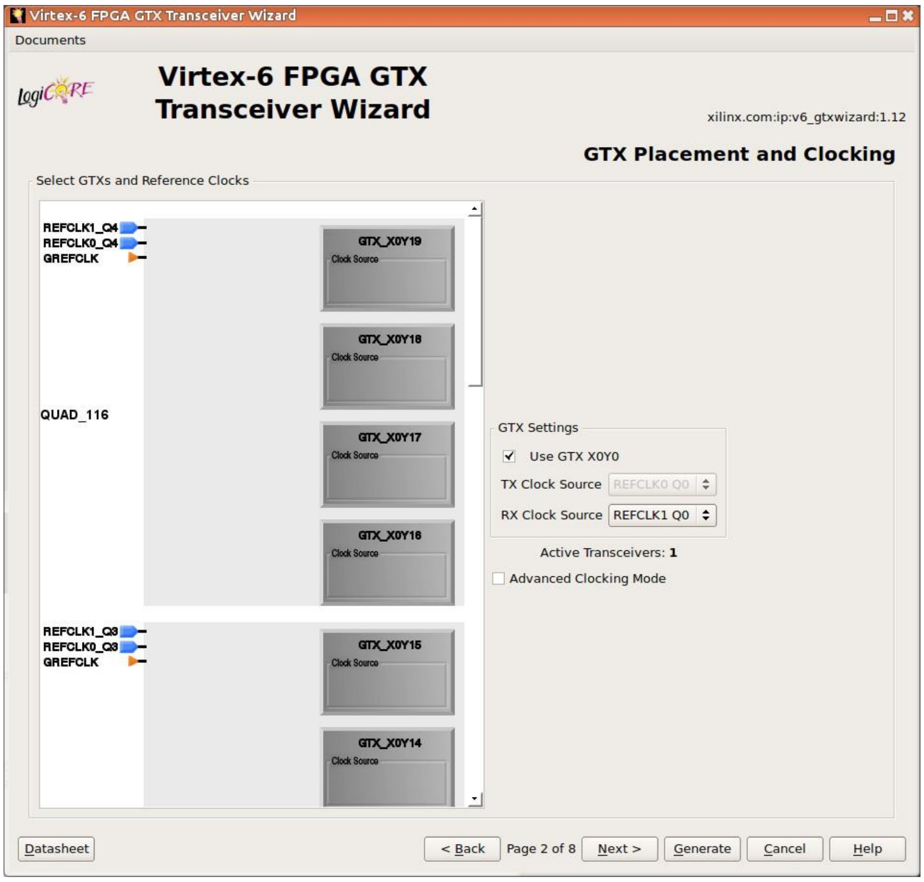
Figure 1: GTX transceiver wizard settings, 1 of 8.

These modules are the GTX wrappers generated by the GTX transceiver Wizard in the Xilinx Core Generator. A step-by-step guide on how to generate these modules and a justification of the Wizard choices can be found next.

2 GTX transceiver Wizard Settings

The GTX transceiver Wizard in the Xilinx Core Generator is necessary in order to generate the wrappers which configure and drive the complex high speed optical transceivers in the FPGA. Version 1.12 is used in this example and choices selected in each of the configuration steps are originated from a mixture of the nature of the MRF protocol along with instructions and recommendations indicated in the Virtex-6 GTX transceiver (UG366 [1]) and the GTX transceiver wizard (UG516 [2]) user manuals.

As currently configured, the embedded receiver acts as receiver only and the transmit path is therefore disabled in the first configuration page (see Fig. 1). The MRF is a custom protocol so no template is available in the Wizard, and the Rx path configuration responds



Virtex-6 FPGA GTX Transceiver Wizard

xilinx.com:ip:v6_gtxwizard:1.12

GTX Placement and Clocking

Select GTXs and Reference Clocks

GTX Settings

[✓] Use GTX X0Y0

TX Clock Source	REFCLK0 Q0
RX Clock Source	REFCLK1 Q0

Active Transceivers: **1**

[ ] Advanced Clocking Mode

Figure 2: GTX transceiver wizard settings, 2 of 8.

to the data rates used in the current version of the hardware. This corresponds to a 2.5 Gbps line rate, 16 bit data path width using 8B/10B encoding, which implies that the 20-bit words are received and decoded at 125 MHz, where the EVR logic receives 16-bit data words at that rate for decoding of the MRF protocol. The `RXCHARISCOMMA` option is selected to indicate the presence of a comma character in the data stream to the rest of the logic. In this case this is a 2-bit wide signal to indicate whether the comma is present in the upper or lower byte of the 16-bit data word each parallel clock cycle.

In Fig. 2 `GTX X0Y0` and `REFCLK1 Q0` are selected as suggested in UG516 for a synchronous application (page 27).

Fig. 3 shows the settings for synchronization and alignment. The `RX buffer` option is enabled even though there is a warning in UG366 (page 319) stating that the elastic buffer is the only non-deterministic block in the receive path. This buffer is there to shift the serial data stream in the case of a misalignment during operation, which can happen under certain conditions when the link is not operating in a fully synchronous mode. This is not the case in the embedded EVR, and the use of this buffer has not been an issue in other applications where the recovered clock is used and divided for use in the FPGA fabric. Disabling the `RX buffer` would imply the need for a custom logic block in the fabric to establish a manual phase alignment mechanism (described in UG366, page 230), which has not been found necessary in other applications. Experiments so far using this configuration in the MRF Embedded EVR have shown a reliable behavior and misalignment of incoming data has not been observed running for long periods of time (several weeks at a time).

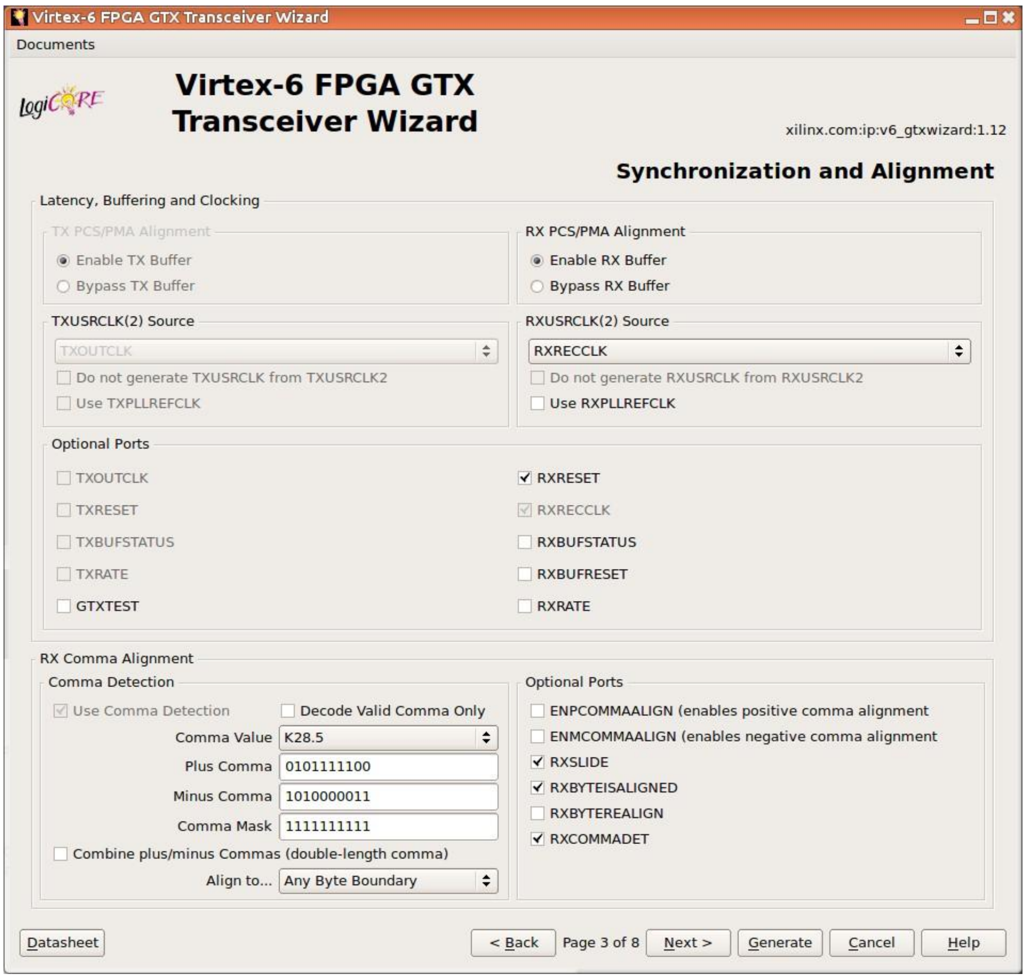
In order to use the divided recovered clock for use in the FPGA fabric, `RXUSRCLK(2)` (RX interface parallel clock) is selected to match the PMA parallel clock (`RXRECCLK`, parallel recovered clock) as shown in the block diagram on page 231 of UG366. The Tx settings are greyed out since it was not enabled in previous configuration screens.

The data alignment mechanism is performed manually in the FPGA fabric as described in UG366 page 219, and alignment ports need to be provided by the wrapper interface. These include `RXRESET, RXSLIDE, RXBYTEISALIGNED, RXCOMMADET`. According to the Xilinx documentation `RXBYTEISALIGNED` should be used in order to monitor the alignment of the data, however experience shows that under this configuration and when cycling over all possible alignment positions using the manual slide mechanism this signal is never asserted. It has therefore been necessary to detect the data alignment in the fabric using the rest of diagnostics provided by the wrapper (see `gtx_byteisaligned.v`).

Settings for the comma detection are selected in order to match the special characters used by the MRF protocol. The comma detection mechanism is enabled to use the transceiver native resources and for the alignment diagnostic ports to be provided to the wrapper interface, however alignment is performed manually as indicated above.

Fig. 4 and Fig. 5 show settings for functions which are mostly not used in this application. The synchronous application option is checked in Fig. 4 and `RXCDRRESET` could be used in order to reset only a part of the GTX transceiver in the manual alignment process, however `RXRESET` is used instead. The loss of sync state machine is enabled in Fig. 5 and the the port `RXLOSSOFSYNCH` is provided to the fabric for diagnostics.

Figs. 6 and 7 are left in the default configuration and no functionality specified there is used. Fig. 8 shows a summary of the GTX transceiver configuration.



Virtex-6 FPGA GTX Transceiver Wizard

xilinx.com:ip:v6_gtxwizard:1.12

Synchronization and Alignment

Latency, Buffering and Clocking

TX PCS/PMA Alignment
- [✓ Enable TX Buffer]
- Bypass TX Buffer

RX PCS/PMA Alignment
- [✓ Enable RX Buffer]
- Bypass RX Buffer

TXUSRCLK(2) Source: TXOUTCLK
- ☐ Do not generate TXUSRCLK from TXUSRCLK2
- ☐ Use TXPLLREFCLK

RXUSRCLK(2) Source: RXRECCLK
- ☐ Do not generate RXUSRCLK from RXUSRCLK2
- ☐ Use RXPLLREFCLK

Optional Ports

☐ TXOUTCLK	✓ RXRESET
☐ TXRESET	✓ RXRECCLK
☐ TXBUFSTATUS	☐ RXBUFSTATUS
☐ TXRATE	☐ RXBUFRESET
☐ GTXTEST	☐ RXRATE

RX Comma Alignment

Comma Detection
- ✓ Use Comma Detection
- ☐ Decode Valid Comma Only

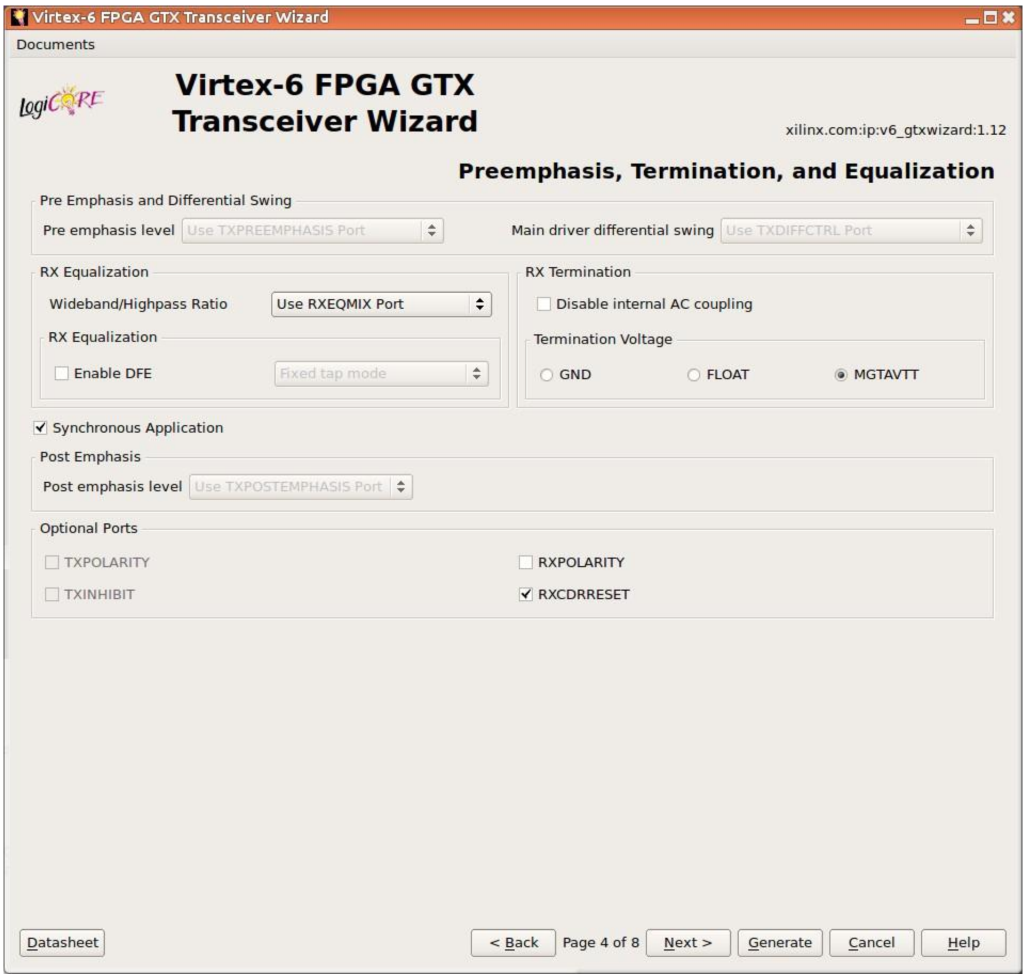
Comma Value	K28.5
Plus Comma	0101111100
Minus Comma	1010000011
Comma Mask	1111111111

- ☐ Combine plus/minus Commas (double-length comma)

Align to... Any Byte Boundary

Optional Ports
- ☐ ENPCOMMAALIGN (enables positive comma alignment
- ☐ ENMCOMMAALIGN (enables negative comma alignment
- ✓ RXSLIDE
- ✓ RXBYTEISALIGNED
- ☐ RXBYTEREALIGN
- ✓ RXCOMMADET

Figure 3: GTX transceiver wizard settings, 3 of 8.



Virtex-6 FPGA GTX Transceiver Wizard

xilinx.com:ip:v6_gtxwizard:1.12

Preemphasis, Termination, and Equalization

Pre Emphasis and Differential Swing

Pre emphasis level: Use TXPREEMPHASIS Port

Main driver differential swing: Use TXDIFFCTRL Port

RX Equalization

Wideband/Highpass Ratio: Use RXEQMIX Port

RX Equalization

☐ Enable DFE — Fixed tap mode

RX Termination

☐ Disable internal AC coupling

Termination Voltage

○ GND ○ FLOAT ● MGTAVTT

☑ Synchronous Application

Post Emphasis

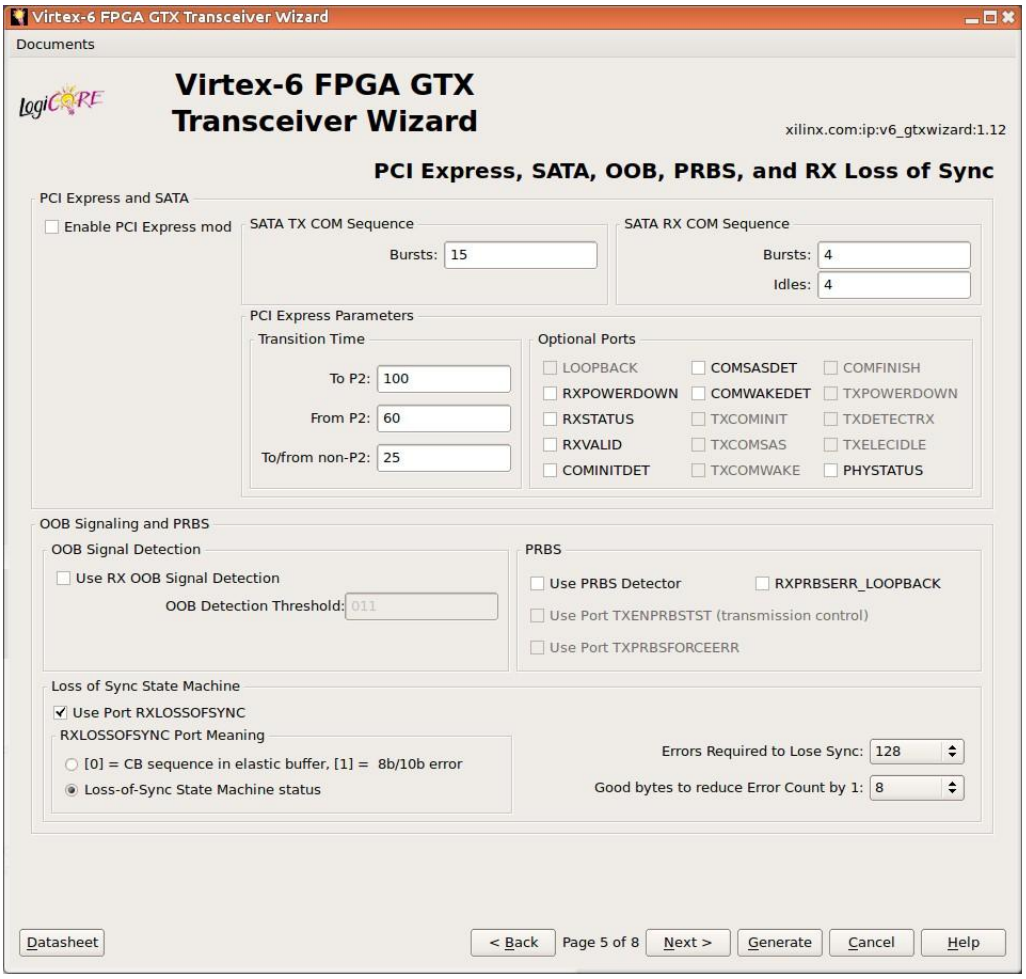
Post emphasis level: Use TXPOSTEMPHASIS Port

Optional Ports

☐ TXPOLARITY ☐ RXPOLARITY

☐ TXINHIBIT ☑ RXCDRRESET

Figure 4: GTX transceiver wizard settings, 4 of 8.



Virtex-6 FPGA GTX Transceiver Wizard

xilinx.com:ip:v6_gtxwizard:1.12

PCI Express, SATA, OOB, PRBS, and RX Loss of Sync

PCI Express and SATA

Enable PCI Express mod

SATA TX COM Sequence

Bursts: 15

SATA RX COM Sequence

Bursts: 4

Idles: 4

PCI Express Parameters

Transition Time

To P2: 100

From P2: 60

To/from non-P2: 25

Optional Ports

LOOPBACK, COMSASDET, COMFINISH, RXPOWERDOWN, COMWAKEDET, TXPOWERDOWN, RXSTATUS, TXCOMINIT, TXDETECTRX, RXVALID, TXCOMSAS, TXELECIDLE, COMINITDET, TXCOMWAKE, PHYSTATUS

OOB Signaling and PRBS

OOB Signal Detection

Use RX OOB Signal Detection

OOB Detection Threshold: 011

PRBS

Use PRBS Detector

RXPRBSERR_LOOPBACK

Use Port TXENPRBSTST (transmission control)

Use Port TXPRBSFORCEERR

Loss of Sync State Machine

[✓ Use Port RXLOSSOFSYNC]

RXLOSSOFSYNC Port Meaning

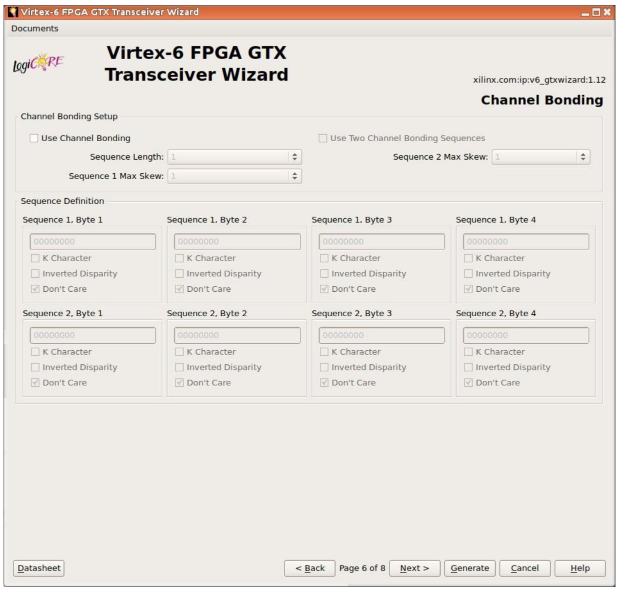
[0] = CB sequence in elastic buffer, [1] = 8b/10b error

[✓ Loss-of-Sync State Machine status]

Errors Required to Lose Sync: 128

Good bytes to reduce Error Count by 1: 8

Figure 5: GTX transceiver wizard settings, 5 of 8.


Virtex-6 FPGA GTX Transceiver Wizard
Documents
Virtex-6 FPGA GTX Transceiver Wizard
xilinx.com:ip:v6_gtxwizard:1.12
Channel Bonding
Datasheet
< Back
Page 6 of 8
Next >
Generate
Cancel
Help

Channel Bonding Setup

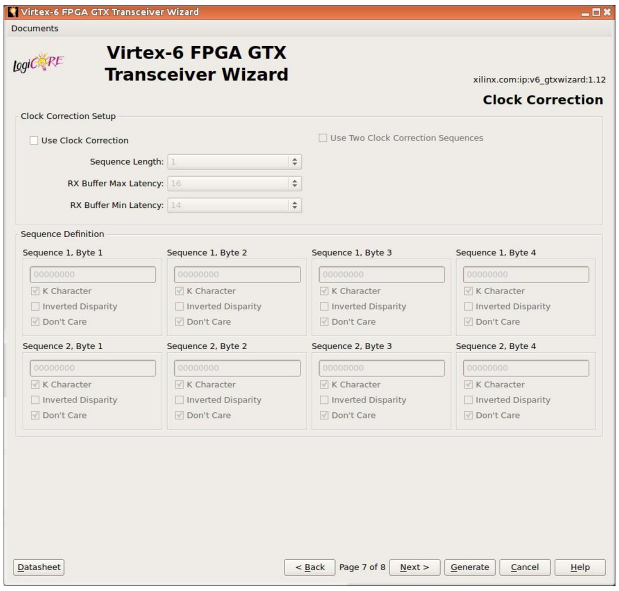
- ☐ Use Channel Bonding
- Sequence Length: 1
- Sequence 1 Max Skew: 1
- ☐ Use Two Channel Bonding Sequences
- Sequence 2 Max Skew: 1

Sequence Definition

	Sequence 1, Byte 1	Sequence 1, Byte 2	Sequence 1, Byte 3	Sequence 1, Byte 4
	00000000	00000000	00000000	00000000
K Character	☐	☐	☐	☐
Inverted Disparity	☐	☐	☐	☐
Don't Care	✓	✓	✓	✓

	Sequence 2, Byte 1	Sequence 2, Byte 2	Sequence 2, Byte 3	Sequence 2, Byte 4
	00000000	00000000	00000000	00000000
K Character	☐	☐	☐	☐
Inverted Disparity	☐	☐	☐	☐
Don't Care	✓	✓	✓	✓

Figure 6: GTX transceiver wizard settings, 6 of 8 (default settings, unused).


Virtex-6 FPGA GTX Transceiver Wizard
Documents
Virtex-6 FPGA GTX Transceiver Wizard
xilinx.com:ip:v6_gtxwizard:1.12
Clock Correction
Datasheet
< Back
Page 7 of 8
Next >
Generate
Cancel
Help

Clock Correction Setup

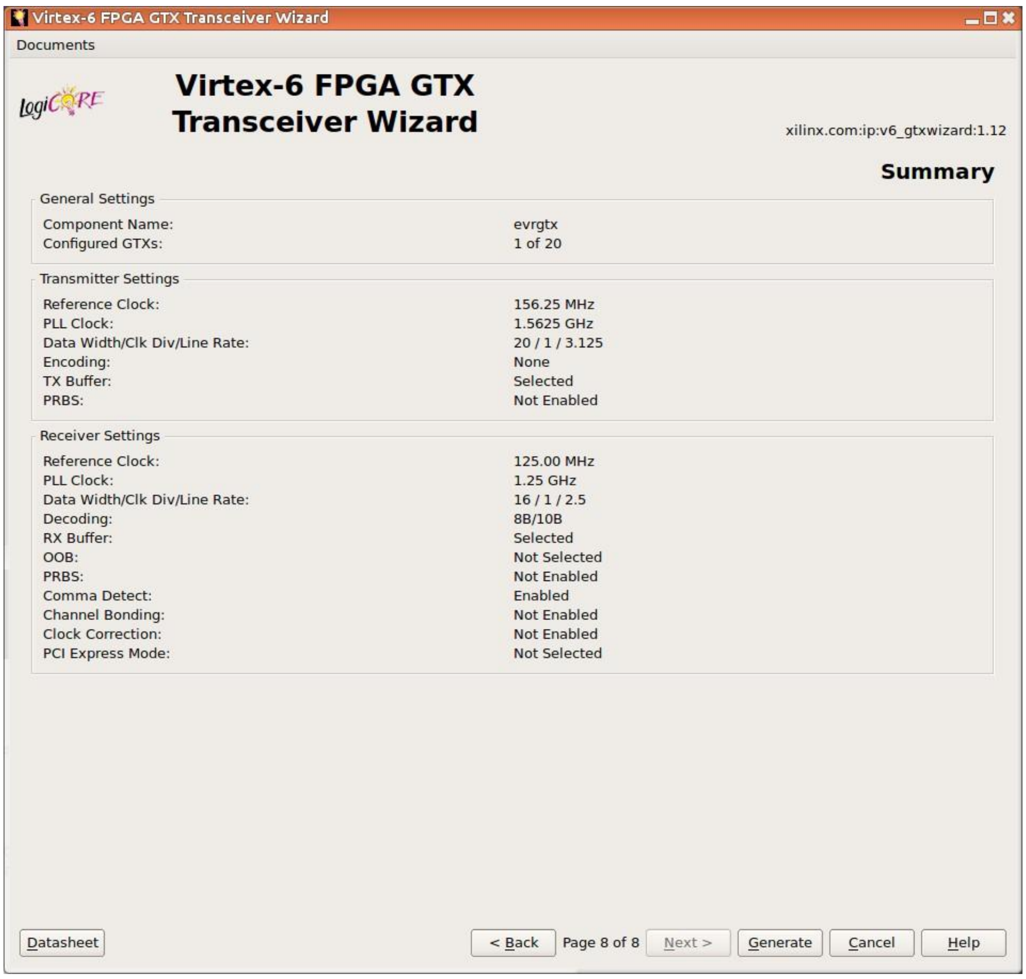
- ☐ Use Clock Correction
- Sequence Length: 1
- RX Buffer Max Latency: 16
- RX Buffer Min Latency: 14
- ☐ Use Two Clock Correction Sequences

Sequence Definition

	Sequence 1, Byte 1	Sequence 1, Byte 2	Sequence 1, Byte 3	Sequence 1, Byte 4
	00000000	00000000	00000000	00000000
K Character	✓	✓	✓	✓
Inverted Disparity	☐	☐	☐	☐
Don't Care	✓	✓	✓	✓

	Sequence 2, Byte 1	Sequence 2, Byte 2	Sequence 2, Byte 3	Sequence 2, Byte 4
	00000000	00000000	00000000	00000000
K Character	✓	✓	✓	✓
Inverted Disparity	☐	☐	☐	☐
Don't Care	✓	✓	✓	✓

Figure 7: GTX transceiver wizard settings, 7 of 8 (default settings, unused).


Virtex-6 FPGA GTX Transceiver Wizard
Documents
Virtex-6 FPGA GTX Transceiver Wizard
xilinx.com:ip:v6_gtxwizard:1.12
Summary
Datasheet
< Back
Page 8 of 8
Next >
Generate
Cancel
Help

General Settings

Component Name:	evrgtx
Configured GTXs:	1 of 20

Transmitter Settings

Reference Clock:	156.25 MHz
PLL Clock:	1.5625 GHz
Data Width/Clk Div/Line Rate:	20 / 1 / 3.125
Encoding:	None
TX Buffer:	Selected
PRBS:	Not Enabled

Receiver Settings

Reference Clock:	125.00 MHz
PLL Clock:	1.25 GHz
Data Width/Clk Div/Line Rate:	16 / 1 / 2.5
Decoding:	8B/10B
RX Buffer:	Selected
OOB:	Not Selected
PRBS:	Not Enabled
Comma Detect:	Enabled
Channel Bonding:	Not Enabled
Clock Correction:	Not Enabled
PCI Express Mode:	Not Selected

Figure 8: GTX transceiver wizard settings, 8 of 8 (summary).

3 Acknowledgements

The base of this work is the BNL's MRF Embedded design, where enhancements to add a deterministic behavior include a module (`gtp_bitslide.v`, converted from VHDL to Verilog using vhd2vl [3]) used in the White Rabbit Switch developed at CERN [4]. The author would like to thank Thomasz Wlostowski from CERN, Larry Doolittle and John Jones from LBNL for discussions on this subject, as well as Jonah Weber and Mike Chin from LBNL for testing the functionality on the NSLS-II BPM test stand.

References

[1] "LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.9,"UG516 (v1.9) March 1, 2011. `http://www.xilinx.com/support/documentation/ip_documentation/v6_gtxwizard/v1_9/ug516_v6_gtxwizard.pdf`

[2] "Virtex-6 FPGA GTX Transceivers", UG366 (v2.6) July 27, 2011. `http://www.xilinx.com/support/documentation/user_guides/ug366.pdf`

[3] "VHDL to Verilog Converter", `http://doolittle.icarus.com/~larry/vhd2vl/`

[4] "The White Rabbit project", `http://www.ohwr.org/projects/white-rabbit`